#dotNET2025

19 de junio

Kinepolis
Ciudad de la Imager
Madrid

Powered by

plain concepts

¿Autenticación sin complicaciones?

#dotNET2025

ORGANIZATION

plain concepts

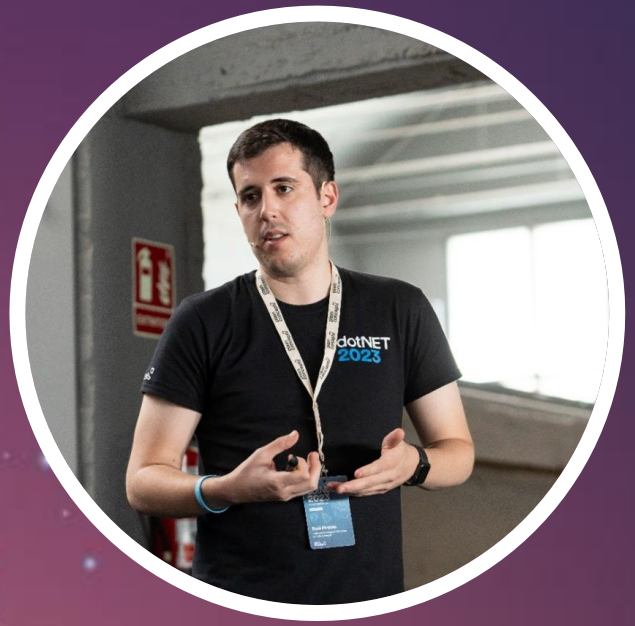
Powered by

plain concepts

NTT DATA Microsoft intel intelequia

dotNET2025
by Plain Concepts

#dotNET2025



Raúl Piracés

Full Stack Engineer @
SCRM International Hub

Víctor Rubio

Software Development Engineer @
Plain Concepts

SAML

OIDC

¿Protocolos de autenticación?

Kerberos

LDAP

¿Por qué OpenID Connect?

Microsoft, Google, Amazon, Okta, Auth0, Apple, GitHub, Salesforce, Twitch

And many more...

OAuth vs OpenID Connect

¿Qué es un token?

- **Varios formatos** posibles…
- **¿Más común? Entonces hablamos de Json Web Token (JWT)**
  - **Otros:** Opacos / por referencia (string random).
- Siempre los encontraremos en la **cabecera "Authorization"**:
  - Authorization: Bearer U2ItcGxIU3RyaW5nQmFzZTY0RW5jb2RIZA==
- **JWT:**
  - **cabecera.payload.firma**
  - En base64 cada "parte" (excepto firma)
  - Utilidades: jwt.io, jwt.ms…

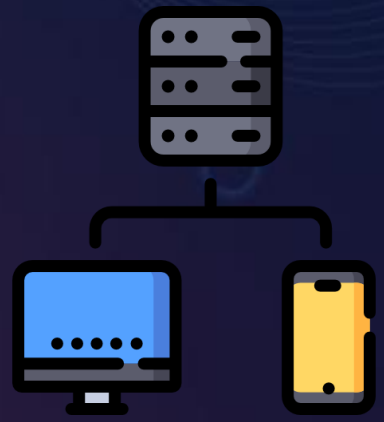
Actores en un flujo de autenticación

- Proveedor de identidad (IdP) – servicio que autentica y emite tokens
- Cliente – aplicación o servicio que solicita la autenticación al IdP
- Usuario final – persona que intenta acceder a un recurso protegido
- Recursos protegidos – datos, servicios que requieren de autenticación (y autorización)

Tipos de aplicaciones cliente

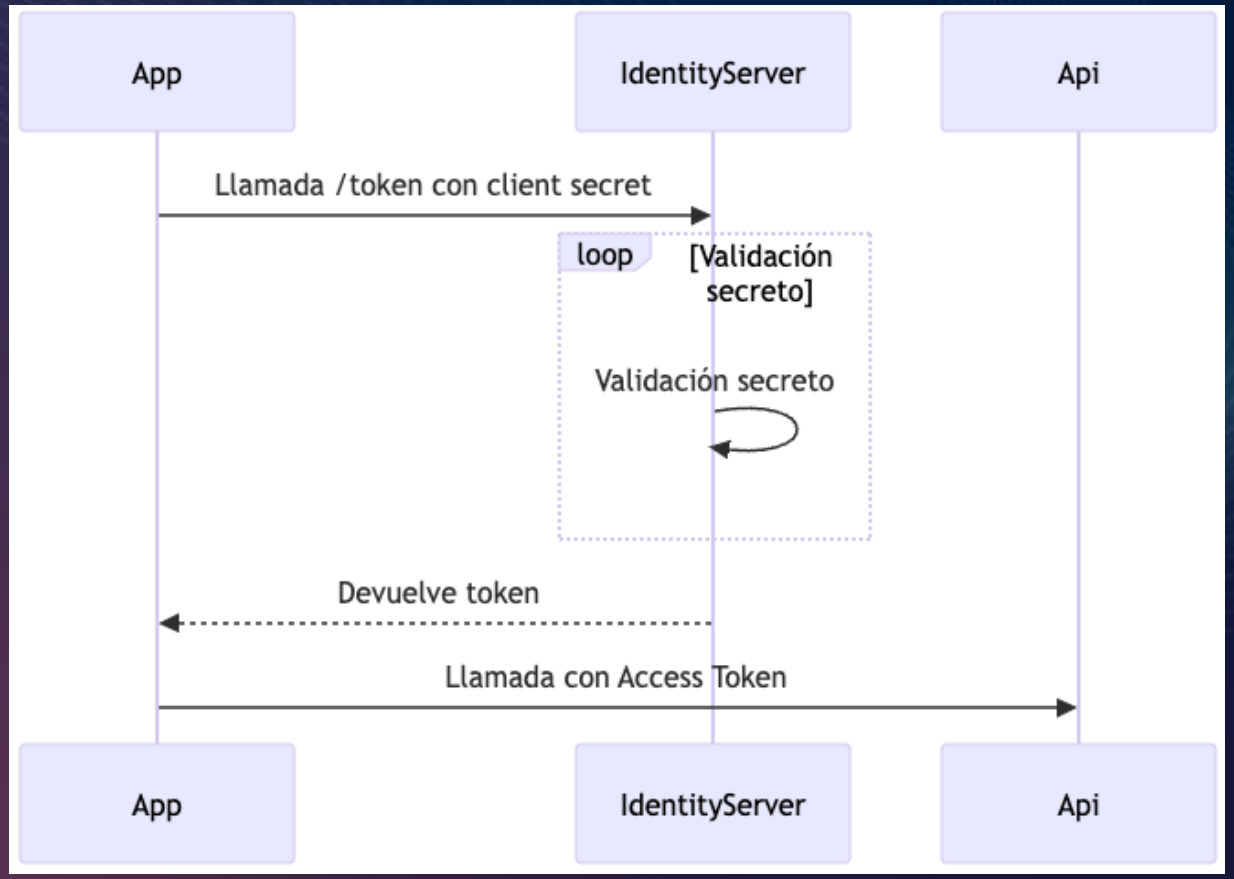
- **No interactivas**: autenticación sin interacción usuario/en nombre de la aplicación
  - **Flujos máquina-máquina, demonios**

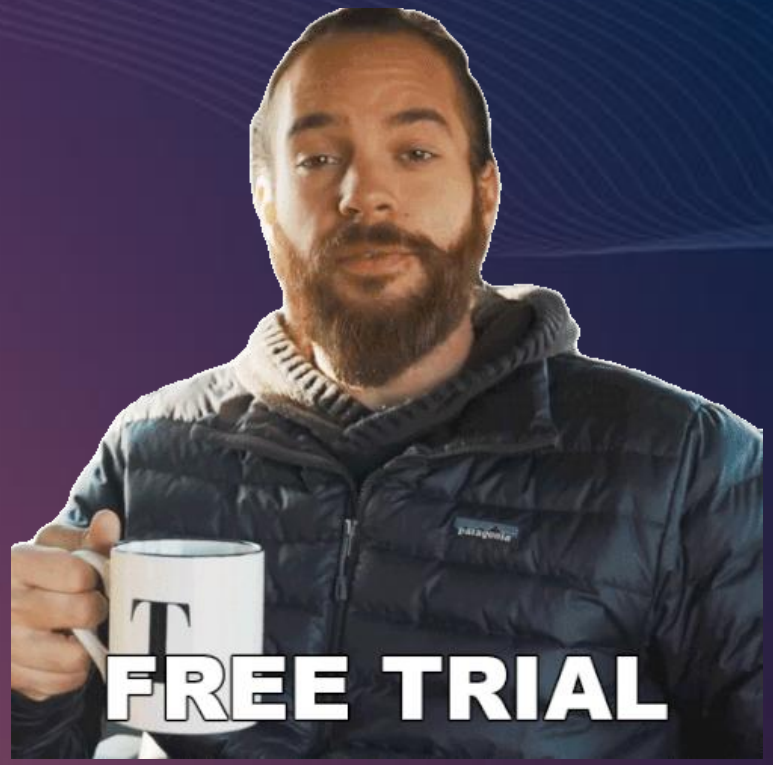
- **Interactivas**: interacción entre el usuario físico y el IdP
  - **SPAs, aplicaciones de escritorio/móviles**



Autenticación cliente no interactivo

- Máquina a máquina (M2M)
- En el estándar:
  - Flujo Client Credentials

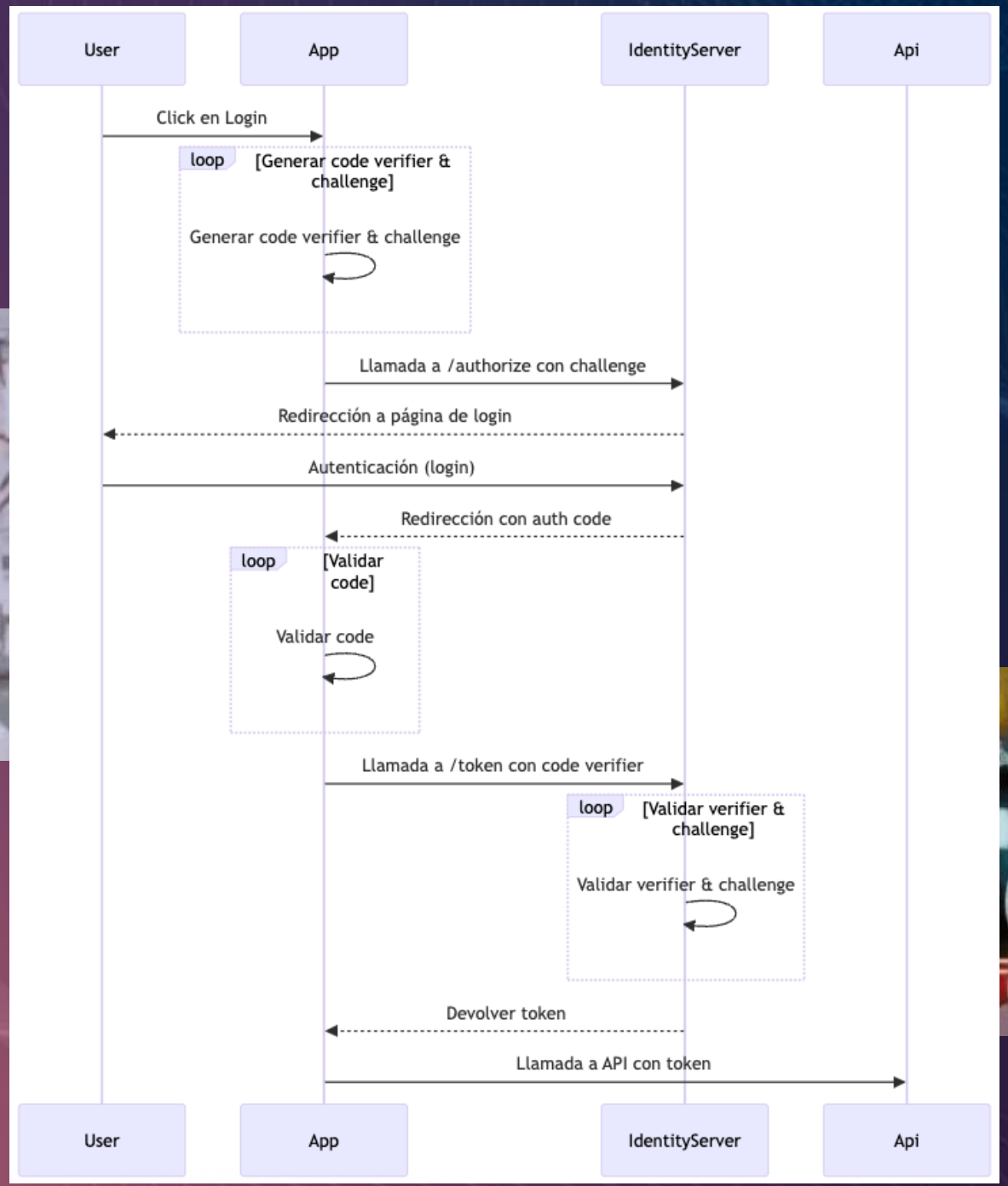


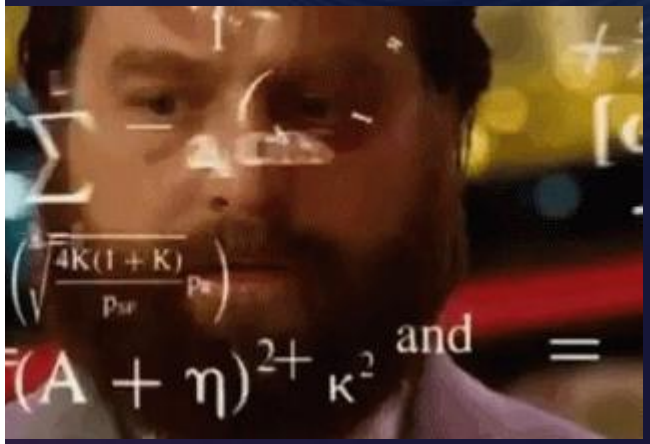


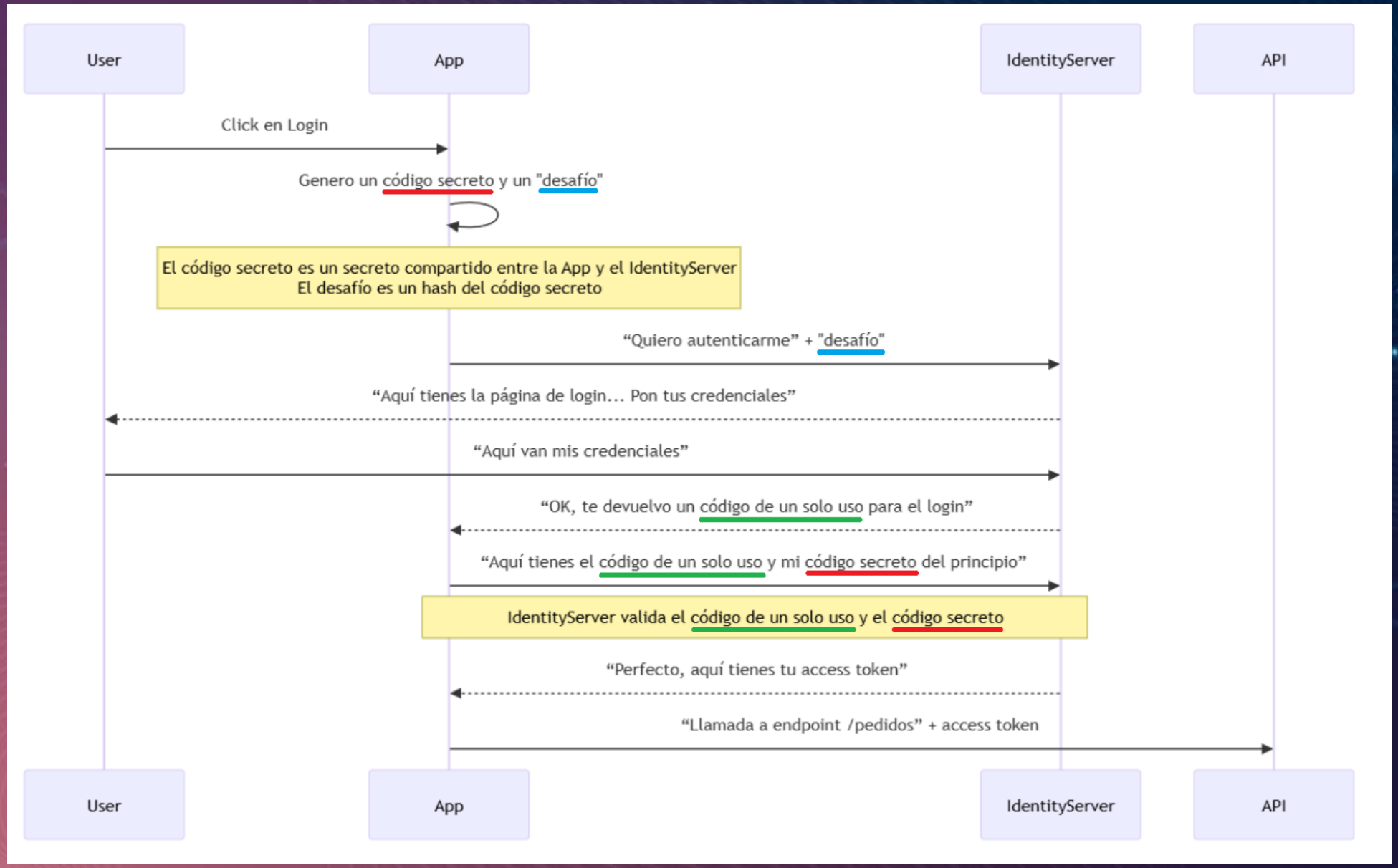
Demo time!

Autenticación cliente interactivo

- Flujo Authorization code + PKCE (obligatorio con OAuth 2.1)
- Credenciales NO expuestas a la aplicación cliente
- Previene ataques de intercepción


User
App
IdentityServer
Api
Click en Login
loop
[Generar code verifier & challenge]
Generar code verifier & challenge
Llamada a /authorize con challenge
Redirección a página de login
Autenticación (login)
Redirección con auth code
loop
[Validar code]
Validar code
Llamada a /token con code verifier
loop
[Validar verifier & challenge]
Validar verifier & challenge
Devolver token
Llamada a API con token
User
App
IdentityServer
Api


and

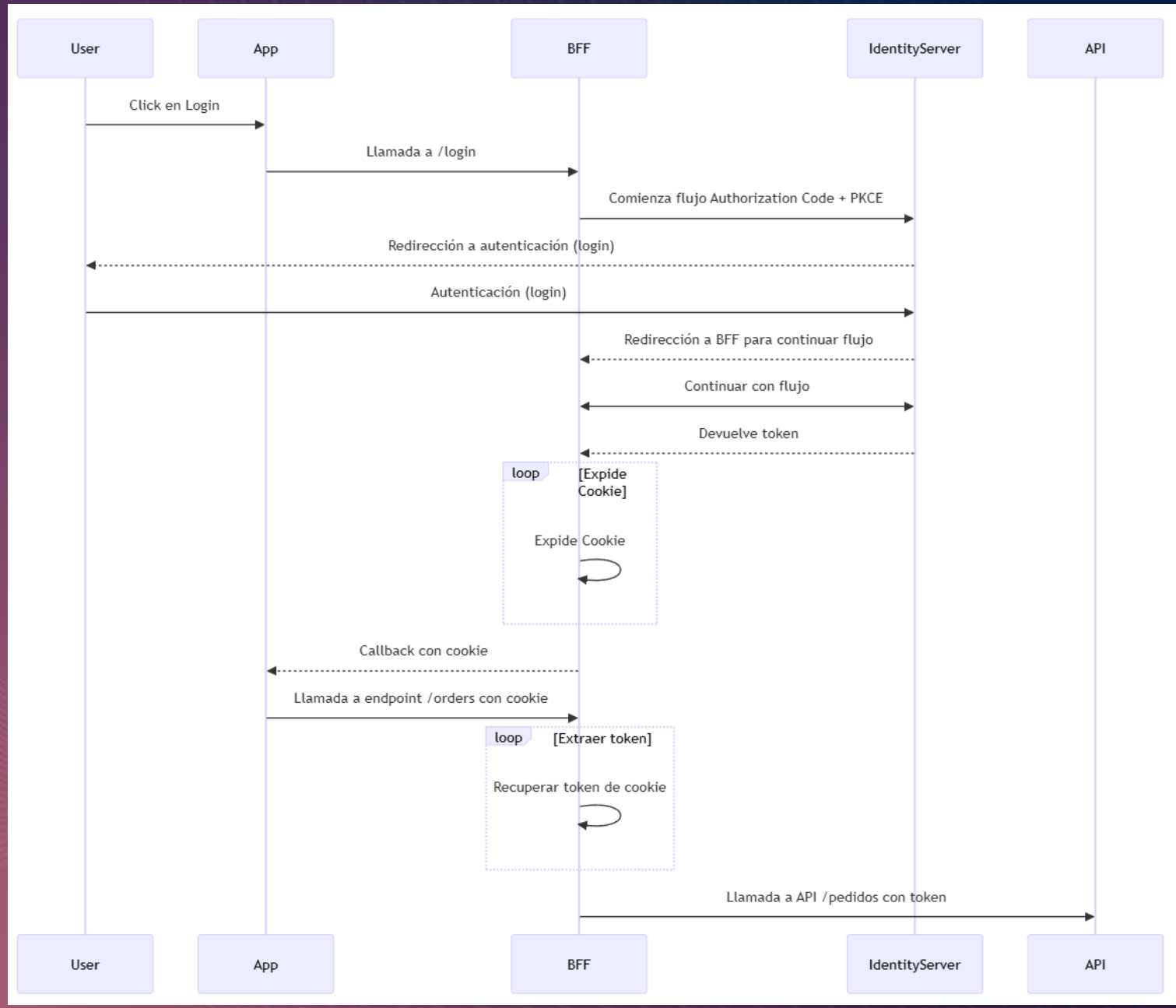

User
App
IdentityServer
API
Click en Login
Genero un código secreto y un "desafío"
El código secreto es un secreto compartido entre la App y el IdentityServer
El desafío es un hash del código secreto
“Quiero autenticarme” + "desafío"
“Aquí tienes la página de login... Pon tus credenciales”
“Aquí van mis credenciales”
“OK, te devuelvo un código de un solo uso para el login”
“Aquí tienes el código de un solo uso y mi código secreto del principio”
IdentityServer valida el código de un solo uso y el código secreto
“Perfecto, aquí tienes tu access token”
“Llamada a endpoint /pedidos” + access token
User
App
IdentityServer
API

Demo time!

¿Preocupaciones?

- Autenticación gestionada en frontend
- Protección del token frente a acceso no autorizado
- Token expuesto al cliente (ataques XSS) + robo de token

Backend for Frontend (BFF)


User
App
BFF
IdentityServer
API
Click en Login
Llamada a /login
Comienza flujo Authorization Code + PKCE
Redirección a autenticación (login)
Autenticación (login)
Redirección a BFF para continuar flujo
Continuar con flujo
Devuelve token
loop
[Expide Cookie]
Expide Cookie
Callback con cookie
Llamada a endpoint /orders con cookie
loop
[Extraer token]
Recuperar token de cookie
Llamada a API /pedidos con token
User
App
BFF
IdentityServer
API

¿Que ha cambiado?

- Responsabilidad de **autenticación en el backend**
- **No se tiene un cliente público** negociando el token (uso de secreto)
- **El frontend no tiene tokens**
- **Sesión basada en cookies**
  - Uso de atributo Same-Site
  - Protección contra XSS - Atributos Secure + HttpOnly
  - Protección contra XSRF
    - Uso de una cabecera custom

Demo time!

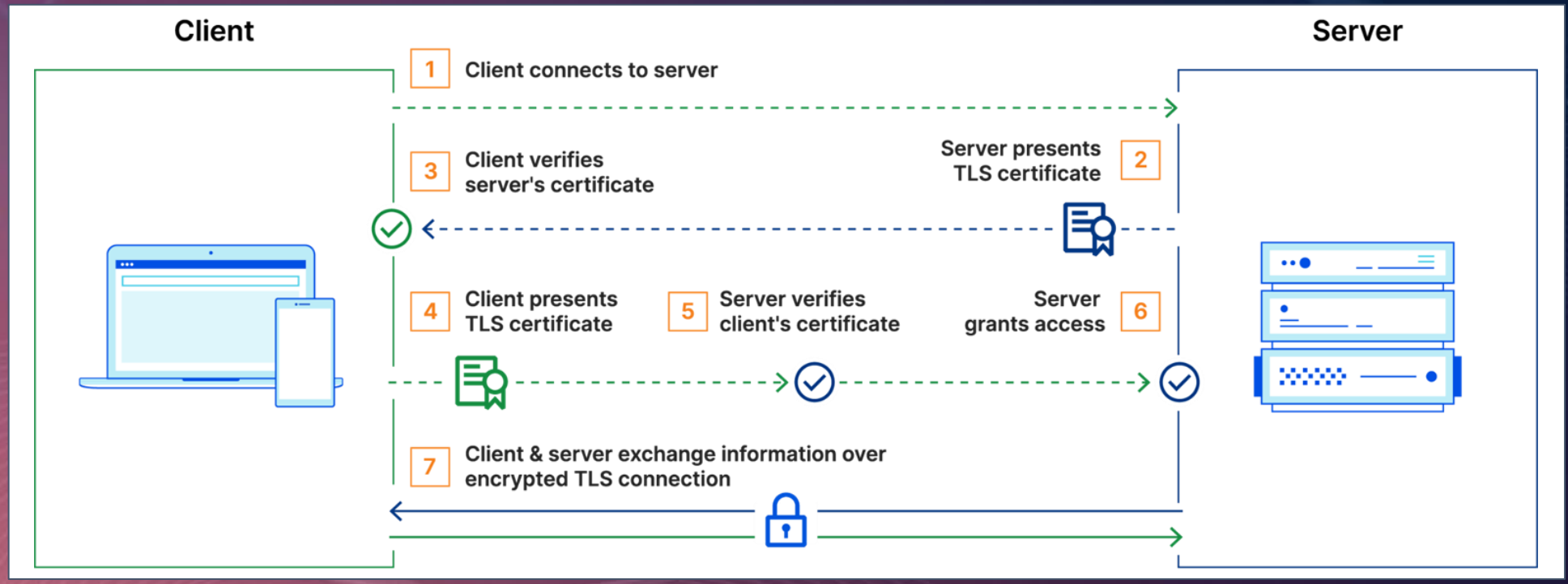
Todavía pueden robar el token...

Pruebas de posesión (POP)

¿PoP?

- Por defecto los tokens en OIDC no se asocian al cliente
  - Riesgo de fuga de tokens
- Los tokens PoP se asocian al cliente que pide el token
  - Uso de criptografía para garantizar que el emisor del token (IdP) es consciente de un secreto del cliente adicional
  - Claim “cnf”
- Uso de Mutual TLS (mTLS) o Demonstrating Proof of Possession (DPoP)

TLS Mutuo

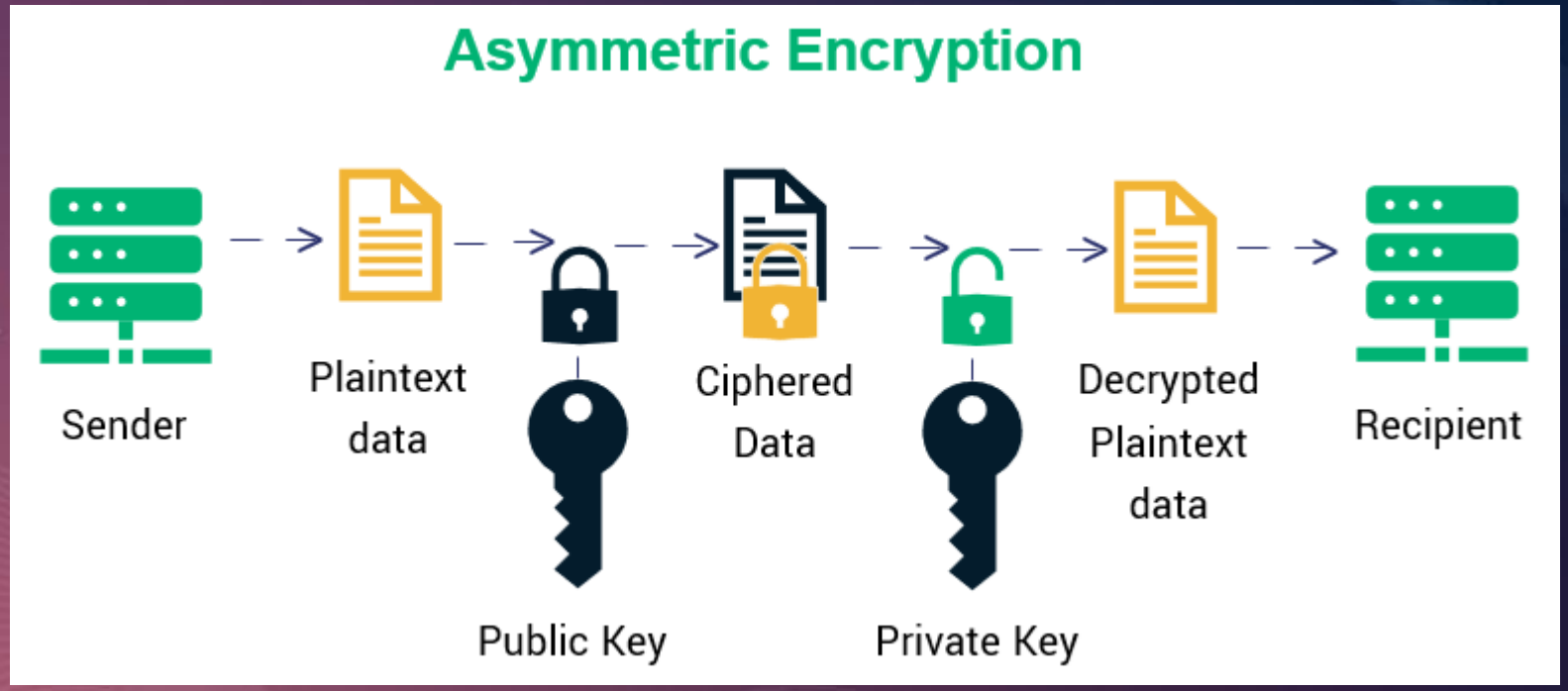

Client
Server
1
Client connects to server
2
Server presents
TLS certificate
3
Client verifies
server's certificate
4
Client presents
TLS certificate
5
Server verifies
client's certificate
6
Server
grants access
7
Client & server exchange information over
encrypted TLS connection

PoP con TLS mutuo

- Asociar un certificado a un Access Token
- Nuevo claim “cnf” que contiene la huella del certificado
- El cliente ha de usar el certificado para llamar a los recursos
- Los recursos protegidos validan el claim “cnf” contra la huella del certificado del cliente

Demo time!

Encriptación asimétrica



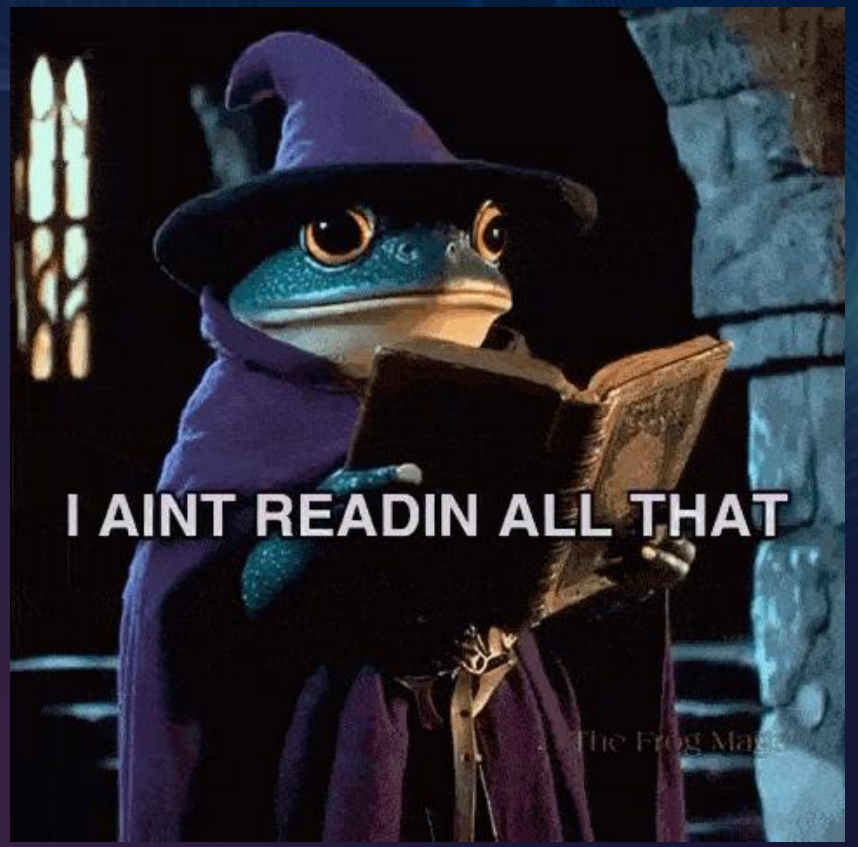
Demonstrating PoP (DPoP)

- Asocia una clave asimétrica a un token de acceso
- El IdP incrustará la huella digital de la clave pública dentro del claim “cnf”
- Al llamar a un recurso protegido, se envía el token de acceso y una nueva prueba DPoP firmada con la clave privada
- **NO** es un mecanismo de autorización

Demo time!

TL;DR

- **Uso de protocolos estándares. OIDC es fácil de implementar y ligero**
- **Dos tipos de clientes**
  - **No interactivos** - Flujo Client Credentials
  - **Interactivos** - Authorization Code + PKCE
- **Mínimo recomendado: Patrón BFF** responsabilidad de autenticación en back y permite sesión basada en cookies
- **PoP permite asociar tokens a un cliente**, minimizando riesgos de fuga de tokens
  - mTLS
  - DPoP



#dotNET2025

plain concepts

dotNET 2025
by Plain Concepts

Thank you!

Powered by plain concepts

NTT DATA Microsoft intel intelequia